Information Sciences 121 (1999) 217–231

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

Fault-tolerant wormhole routing in two-dimensional mesh networks with convex faults

Pao-Hwa Sui, Sheng-De Wang [*]

Department of Electrical Engineering, EE building, National Taiwan University, Room 441, 1 Roosevelt Road, Sec. 4, Taipei 106, Taiwan, ROC

Received 1 June 1996; received in revised form 3 November 1998; accepted 4 May 1999

Abstract

Faulty blocks are expanded, by disabling nodes, to form rectangular faults in many existing works to facilitate the designing of deadlock-free routing schemes for meshes recently. In this paper, faulty blocks are diffused to be rectangular faults, which are composed of faulty and fault-diffused nodes. These rectangular faults are then shrunk, by recovering fault-diffused nodes, to form convex faults for reducing the number of nodes disabled. Simulation results show that up to 70% of the disabled nodes, which are needed to form rectangular faults, can be recovered if the number of faulty nodes is less than 10% of the total network nodes.

Both non-adaptive and adaptive fault-tolerant routing algorithms are proposed to handle these resulted convex faults. The adaptive routing algorithm is enhanced from the non-adaptive counterpart by utilizing the virtual channels that are not used in the non-adaptive algorithm; hence, the number of virtual channels per physical channel used in the adaptive algorithm is the same to that used in the non-adaptive algorithm. © 1999 Published by Elsevier Science Inc. All rights reserved.

1. Introduction

Direct networks have become a popular means for interconnecting components of massively parallel computer systems. In direct networks, nodes

[*] Corresponding author. Fax: +886-2-23671909.
E-mail address: sdwang@cc.ee.ntu.edu.tw (S.-D. Wang).

(computers) are connected to only a few nodes, its neighbors, according to the topology of the network and communicate with each other by passing messages. The *n*-dimensional mesh (*n*-D mesh) network is currently the most popular topology for massively parallel computer systems. Low-dimensional mesh networks, due to their low node degree, are more popular than the high-dimensional mesh networks. The two-dimensional mesh (2-D mesh) topology has been adopted by Symult 2010 [4], Intel Touchstone DELTA [5] and Intel paragon; the MIT J-machine adopts three-dimensional mesh topology.

The *wormhole* switching technique by Dally and Seitz [3] has been widely used in contemporary multiprocessors for exchanging messages. *Virtual-cut-through* [1] and *store-and-forward* [2] are the other two often used switching techniques. In the wormhole switching, a message is divided into packets and a packet is composed of flow control digits or *flits*. The header flit governs the route. As the header advances along a specific route, the remaining flits follow in a pipeline fashion. If the header encounters a busy channel, it is blocked until the channel become available and all the flits in same packet remain in the flit buffer along the specified route. A survey of wormhole routing for direct networks can be found in [8]. Routing schemes proposed in this paper are based on wormhole switching.

An interconnection network cannot avoid having failure components in real world. Simulating multiple virtual channels on each physical channel has been used to support fault-tolerance and adaptability [6,7,9–12,14]. Adaptive routing can tolerate faults by selecting message paths but the fault-tolerance is not guaranteed.

Linder and Harden [6] extend the concept of virtual channel to multiple virtual interconnection networks that provide adaptability and fault-tolerance. Messages routing in a virtual network are constrained to travel the virtual networks in a pre-defined order. The main disadvantage of their method is that the number of virtual channels per physical channel needed is large.

Chien and Kim [7] present a partially adaptive algorithm for mesh networks. Only three virtual channels per physical channel are required in their methods. Faults are expanded to be rectangular, convex in [7], faults to ensure the correctness of their algorithms. Extra nodes are disabled when faults are located on the boundaries of meshes. For instance, even only one node or link fault on a boundary row of a two-dimensional mesh, all nodes on that row must be disabled.

Dally and Aoki [9] present two adaptive routing algorithms for direct networks. Both algorithms, based on the concept of dimension reversal, are deadlock-free by eliminating cycles in the resource dependency graph. Faults are not expanded to be rectangular faults. However, the number of faults tolerated depends on the number of virtual channels used, in the static algorithm, and depends on the locations of the faults, in the dynamic algorithm.

Boppana and Chalasani [10–12] present techniques to enhance existing routing algorithms for fault-tolerance in meshes with multiple rectangular faults. The concepts of f-rings and f-chains are introduced and are used for routing messages around rectangular faults. Two, three or four virtual channels are required for different fault situations. *Arbitrarily located* rectangular faults are considered in [12] but not in [10,11]. Although Boppana and Chalasani claim that their methods in [12] provide deadlock-free message routing in two-dimensional meshes, deadlocks among column messages may occur under some fault situation [13].

Su and Shin [14] present fault-tolerant routing algorithms, which decompose the network into two virtual interconnection networks, VIN1 and VIN2. VIN1 supports deadlock-free routing and VIN2 support fully adaptive routing. VIN2 trades its adaptability for fault-tolerance when the requested channels in VIN1 are not connected to safe nodes. For mesh networks, faulty blocks are expanded to *disconnected* rectangular blocks, which are composed of unsafe and faulty nodes. Two rectangular blocks, B_1 and B_2, are disconnected, if for each node $x_1 \in B_1$ and $x_2 \in B_2$, the distance between x_1 and x_2 in at least one-dimension is no smaller than three. Expanding faulty blocks to be *disconnected* rectangular faults may cause even more nodes disabled than just expanding faulty blocks to be rectangular faults. Although only two virtual channels per physical channel are needed in their method, deadlocks can also occur under some fault situation [17].

In all the routing methods mentioned above, either the number of faults tolerated is up to the number of virtual channels used [9] or the faults are restricted to or expanded to be rectangular faults [7,10–12,14]. Simulating multiple virtual channels on every physical channel is not a bargain without cost. Large number of virtual channels per physical channel causes the increase of hardware complexity, cost and delay in routing logic. Expanding faults to be rectangular faults causes good nodes unnecessarily disabled. The more the good nodes were disabled, the less the computing and transmission power was left. In [14], though not all nodes in the expanded rectangular fault are disabled, only corner nodes of each rectangular fault could be in unsafe state if they are not faulty. These unsafe nodes can transmit and receive but not relay messages. In this paper, for the purpose of reducing the number of nodes disabled, each expanded rectangular fault is shrunk, by recovering nodes, to form one or many convex faults. A non-adaptive fault-tolerant routing algorithm is first proposed. Only four virtual channels per physical channel are required in the algorithm. The non-adaptive algorithm is then enhanced to be an adaptive algorithm by utilizing the virtual channels that are not used in the non-adaptive algorithm. Since the adaptability is introduced by utilizing the virtual channels that are not used in the non-adaptive counterpart, the same number of virtual channels per physical channel are used in these two algorithms. The rest of this paper is organized as follows: Section 2 describes terminology. The methods for diffusing faults and shrinking rectangular faults are also described in Section 2. Simulation results of the fault shrink approach are also given in this section. A non-adaptive routing algorithm is proposed in Section 3. The non-adaptive algorithm is enhanced in Section 4 to be an adaptive algorithm. This paper concludes with Section 5.

2. Preliminaries

An n-dimensional mesh has $k_{n-1}, k_{n-2}, \ldots, k_0$ nodes, k_i nodes along dimension i, $0 \leqslant i \leqslant n-1$, and $k_i \geqslant 2$. Each node x is uniquely indexed by an n-tuple $(x_{n-1}, x_{n-2}, \ldots, x_0)$, where $0 \leqslant x_i \leqslant k_i - 1$. Nodes $x = (x_{n-1}, x_{n-2}, \ldots, x_0)$ is a positive (resp. negative) neighbor to node $y = (y_{n-1}, y_{n-2}, \ldots, y_0)$ if and only if $x_i = y_i$ for all i except one, j, where $x_j = y_j + 1$ (resp. $= y_j - 1$). Each node has from n to $2n$ neighbors up to their location on the mesh. Neighboring nodes are connected by a direct link implemented by two unidirectional physical channels with opposite directions. When node x is a positive (resp. negative) neighbor of node y, the channel from node y to node x is a positive (resp. negative) channel. The four directions of a two-dimensional mesh are labeled, hereinafter, as North, East, South and West.

A faulty block is a set of connected faulty nodes and links. The fault boundary, f-boundary, of a faulty block is the set of good nodes and links immediately surrounding the faulty block. Faulty blocks in any shape can be diffused to be rectangular faults, whose f-boundaries are in shapes of rectangles. A faulty block is a convex fault if there are no good nodes located outside the block are surrounded by the fault in the two opposite directions of a dimension. A node is a corner node of an f-boundary if it is on the f-boundary and is located at distance two to the fault surrounded by the f-boundary.

2.1. *Fault-diffusion*

Fault-diffusion: Each good node that has two or more faulty or fault-diffused links in different dimensions set itself to be a fault-diffused node.

Each node is in good, fault-diffused or faulty state after fault-diffusion. Links incident to fault-diffused nodes are fault-diffused links. All fault-diffused nodes can be identified recursively. Fig. 1 shows an 8×8 mesh with faulty nodes (shown as filled circles). After fault-diffusion, nodes a, b, c, d, e, f, g, h, i, j, k, l and m are set to be fault-diffused nodes and two rectangular faults are formed. Heavy lines in Fig. 1 indicate f-boundaries and dotted lines indicate fault-diffused links. It is possible that more than one faulty blocks are contained in a rectangular fault after fault-diffusion.

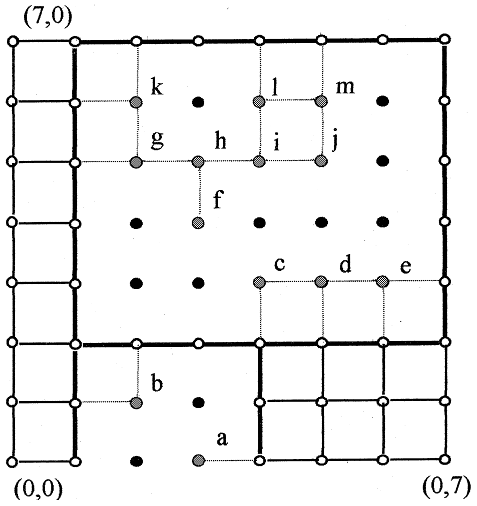

Fig. 1. An example result of fault-diffusion.

2.2. Fault-shrink

Fault-shrink: Each fault-diffused node that has a good positive (resp. negative) neighbor on dimension i, $0 \leqslant i \leqslant 1$, generates and sends a flag, $f1$, to its negative (resp. positive) neighbor on dimension i. Flag $f1$ is propagated without changing direction until a fault node is reached. Any fault-diffused node that receives and/or generates two $f1$ flags is recovered to be a good node. Each recovered node then generates and sends a flag, $f2$, to its fault-diffused neighbors along the direction that is opposite to the direction on which $f1$ received. Flag $f2$ is propagated without changing direction until a good node is reached. Any fault-diffused node that accepts one or more $f2$ flags is also recovered to be a good node. A fault-diffused node that is not recovered sets itself to be a faulty node after the fault-shrink process.

Each rectangular fault may become several convex faults after fault-shrink. Fig. 2 is the final result of Fig. 1 after fault-shrink. Nodes b, c, d, e, g, i, j and k are recovered due to the receiving of flag $f1$ and nodes h, l and m are recovered due to the receiving of flag $f2$. Three convex faults are formed in Fig. 2.

To each f-boundary, the only set of connected nodes which has largest index value on dimension 0 (resp. dimension 1) with respect to all other nodes on the f-boundary is called the EMAX (resp. NMAX) of the f-boundary; the only set of connected nodes which has smallest index value on dimension 0 (resp. dimension 1) with respest to all other nodes on the f-boundary is called the EMIN (resp. NMIN) of the f-boundary. We assume that the length of each

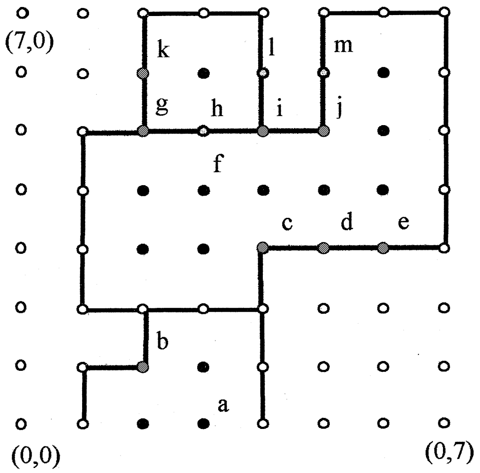

Fig. 2. Result of Fig. 1 after fault-shrink.

convex fault is smaller than its corresponding dimension length, for a whole row or column failure disconnects two-dimensional meshes.

Lemma 1. *Only convex faults exist after fault-shrink.*

Proof. Lemma 1 is proved by showing that no good node is surrounded by the same fault in the two opposite directions of a dimension. Good nodes are classified into three disjointed classes. Class A contains nodes that are never contained in rectangular fault. Class B contains nodes that are recovered due to the receiving of flag $f1$ and class C contains nodes that are recovered due to the receiving of flag $f2$.

Since all faulty blocks are rectangular faults after fault-diffusion, each node in class A has at most one link connects to a rectangular fault. Therefore, no node in class A can be surrounded by same faulty block in the two opposite directions of a dimension. Each node in class B can reach f-boundaries, which are composed of good nodes and links, in two directions of different dimensions; hence, it cannot be surrounded by same fault in the two opposite directions of a dimension. A rectangular fault is decomposed, by nodes of classes B and C, into several convex faults if there are fault-diffused nodes recovered due to the receiving of flag $f2$. Therefore, the two convex faults located at the two opposite sides of nodes in class C are disconnected and, hence, no node in class C is surrounded by a fault in the two opposite directions of a dimension. From above arguments, no good nodes are surrounded by a fault in the two opposite directions of a dimension, hence only convex faults exist after fault-shrink. □

2.3. Simulation results

In order to evaluate how many percentages of the fault-diffused nodes can be recovered by fault-shrink, we simulated a 16×16 mesh with different percentages of the total network nodes being faulty. Symbols F and N denote the total number of faulty and network nodes. Fault-diffusion and fault-shrink are performed each time after the required faulty nodes are generated randomly. Table 1 shows N_d, the total number of fault-diffused nodes, N_{r1}, the number of nodes recovered due to the receiving of flag $f1$, and N_{r2}, the number of nodes recovered due to the receiving of flag $f2$, of 1000 experiments in each of the different fault situations. The last column of Table 1 indicates how many percentages of the fault-diffused nodes are recovered and shows that the smaller the percentage of the total network nodes being faulty, the larger the percentage of the fault-diffused nodes can be recovered. Upto 70% of the fault-diffused nodes are recovered when the number of total faulty nodes is less than 10% of the total network nodes.

3. Non-adaptive fault-tolerant routing

The following notations are defined to facilitate our discussion:
Destination node $d = (d_1, d_0)$.
Current node $c = (c_1, c_0)$.
Routing tag $R = (r_1, r_0) = (d_1 - c_1, d_0 - c_0)$.
$f(R)$: number of non-zero elements in R.
$v(R, i)$: the value of the ith element in R, $0 \leqslant i \leqslant 1$.
$\mathrm{VC}_{i,\pm j}$: the virtual channel of class j, $0 \leqslant j \leqslant 3$, of dimension i, $0 \leqslant i \leqslant 1$; $\mathrm{VC}_{i,+j}$ (resp. $\mathrm{VC}_{i,-j}$) represents positive (resp. negative) virtual channel.

3.1. Non-adaptive fault-tolerant routing algorithm

The non-adaptive fault-tolerant routing algorithm proposed is based on the e-cube algorithm [3]. Messages are routed firstly to the columns on which their

Table 1
Simulation results

F/N (%)	N_d	N_r1	N_r2	$(N_\mathrm{r1} + N_\mathrm{r2})/N_\mathrm{d}$
1	75	69	0	0.92
5	2474	1968	36	0.81
10	14 623	9588	894	0.72
15	58 092	17 573	4712	0.38
20	139 734	10 635	3587	0.10
25	175 073	2892	1205	0.02

destination nodes reside and then routed to their destinations. A message that has not yet reached the column on which its destination resides is a row message; a message that is not a row message is a column message. Row messages traveling from West to East (resp. East to West) are WE (resp. EW) messages. Similarly, column messages are classified into NS and SN messages. Since each message is routed firstly to the column on which its destination resides and then routed to its destination, a row message may become a column message but a column message never becomes a row message.

Algorithm 3.1. non-adaptive fault-tolerant routing algorithm for two-dimensional meshes with convex faults.

```
1. Calculate $R$;
2. If $R = 0$, route the message to the local processor and exit;
3. If the message is a WE (resp. EW) message
     then if the requested $VC_{0,+0}$ (resp. $VC_{0,-0}$) is not connected to a fault
       then request $VC_{0,+0}$ (resp. $VC_{0,-0}$)
       else request $VC_{1,+1}$ (resp. $VC_{1,+2}$), if $v(R,1) \geqslant 0$, or $VC_{1,-1}$ (resp. $VC_{1,-2}$),
       if $v(R,1) \leqslant 0$, until the corner node is reached. /* If the mesh boundary is
       reached or the requested virtual channel of dimension 1 is faulty, then re-
       verse the direction */
4. If message is an NS (resp. SN) message
     then if the requested $VC_{1,-0}$ (resp. $VC_{1,+0}$) is not connected to a fault
       then request $VC_{1,-0}$ (resp. $VC_{1,+0}$)
       else route message on the $f$-boundary in direction of clockwise until the
       corresponding node on dimension 1 is reached. /* If the mesh boundary
       is reached or the requested $VC_{0,+1}$ (resp. $VC_{0,-2}$) is faulty, then reverse the
       direction to be counter-clockwise */
```

WE (resp. EW) messages request $VC_{0,+0}$ (resp. $VC_{0,-0}$) if they are not blocked by faults. If the requested virtual channel is connected to a fault, the WE (resp. EW) message requests $VC_{1,+1}$ (resp. $VC_{1,+2}$), if $v(R,1) \geqslant 0$, or $VC_{1,-1}$ (resp. $VC_{1,-2}$), if $v(R,1) \leqslant 0$, until the corner node is reached. The message performs a U-turn and reverses its direction if the mesh boundary is reached or the requested virtual channel of dimension 1 is connected to a fault before the corner node is reached. WE (resp. EW) messages request $VC_{0,+0}$ (resp. $VC_{0,-0}$) again at the corner nodes.

NS (resp. SN) messages requests $VC_{1,-0}$ (resp. $VC_{1,+0}$) if they are not blocked by faults. If the requested virtual channel is connected to a fault at node x, the column message is then routed in the direction of clockwise along the f-boundary until the corresponding node on dimension 1 of x is reached. The corresponding node on dimension 1 of $x = (x_1, x_0)$, which has a positive (negative) faulty channel on dimension 1, is the node $y = (y_1, y_0)$, which has a negative (positive) faulty channel on dimension 1, located on the other side of the *f*-boundary of the same fault, where $y_1 \neq x_1$ and $y_0 = x_1$. NS (resp. SN) messages use $VC_{0,+1}$, $VC_{1,-0}$, $VC_{0,-1}$ and $VC_{1,+3}$ (resp. $VC_{0,-2}$, $VC_{1,+0}$, $VC_{0,+2}$ and $VC_{1,-3}$) as they traveling in the corresponding direction along the *f*-boundary in direction of clockwise. If the mesh boundary is reached or the requested $VC_{0,+1}$ (resp. $VC_{0,-2}$) is connected to a fault, the message performs a U-turn and reverses its direction to counter-clockwise. NS (resp. SN) messages use $VC_{0,-3}$, $VC_{1,-0}$, $VC_{0,+1}$, and $VC_{1,+3}$ (resp. $VC_{0,+3}$, $VC_{1,+0}$, $VC_{0,-2}$ and $VC_{1,-3}$) as they traveling in the corresponding direction along the *f*-boundary in direction of counter-clockwise. NS (resp. SN) messages request $VC_{1,-0}$ (resp. $VC_{1,+1}$) again at the corresponding node on dimension 1 of node x.

3.2. Usage of virtual channels

The usage of virtual channels by different types of messages is depicted in Fig. 3. It is noticed that the sets of virtual channels used by different types of messages are disjointed. Heavy lines in Fig. 3 indicate the mesh boundary or *f*-boundaries.

WE messages use $VC_{0,+0}$, $VC_{1,+1}$ and $VC_{1,-1}$ and EW messages use $VC_{0,-0}$, $VC_{1,+2}$ and $VC_{1,-2}$ as they traveling in the corresponding directions. NS messages use $VC_{1,-0}$, $VC_{0,+1}$, $VC_{1,+3}$ and $VC_{0,-1}$ as they traveling in the corresponding directions with the exception that an NS message whose routing direction on the *f*-boundary has been set to counter-clockwise uses $VC_{0,-3}$ while traveling from East to West. SN messages use $VC_{1,+0}$, $VC_{0,+2}$, $VC_{1,-3}$ and $VC_{0,-2}$ as they traveling in the corresponding directions with the exception that

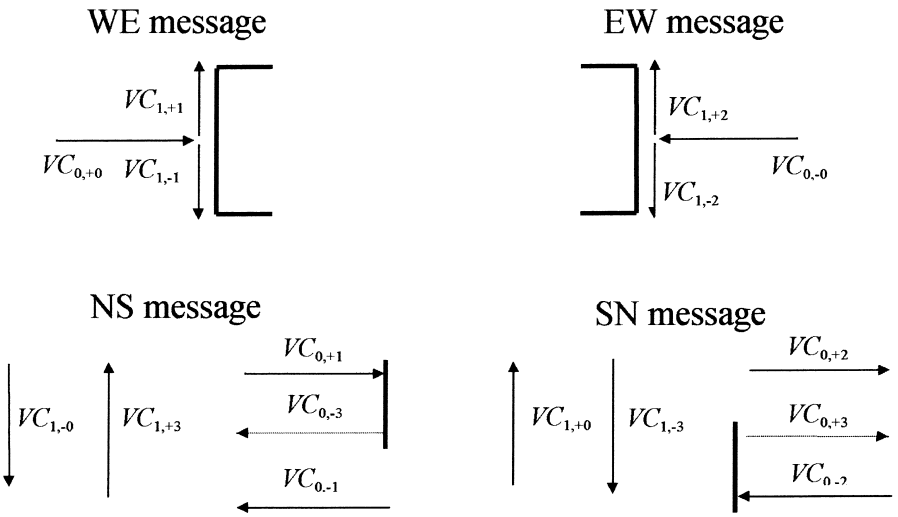

Fig. 3. Usage of virtual channels.

an SN message whose direction has been set to counter-clockwise uses $VC_{0,+3}$ while traveling from West to East.

3.3. Example

Let us consider the example of routing a message from node $a = (1,1)$ to node $k = (6,4)$ in Fig. 4. The message is routed as a WE message from node a to node b using $VC_{0,+0}$. Since the requested $VC_{0,+0}$ is faulty and $v(R,1) = 5$ at node b, the message then requests $VC_{1,+1}$ until the corner node c is reached. At node c, the message requests $VC_{0,+0}$ again until node d, at which the message becomes an SN message, is reached. The message then requests $VC_{1,+0}$ at node d. Since the requested $VC_{1,+0}$ is faulty and $VC_{0,-2}$ is also faulty at node e, the message is routed on the *f*-boundary in counter-clockwise until node j, the corresponding node of e, is reached. $VC_{0,+3}$ is used from node e to node f; $VC_{1,+0}$ is used from node f to node g; $VC_{0,-2}$ is used from node g to node h; $VC_{1,-3}$ is used from node h to node i and $VC_{0,-2}$ is used from node i to node j. At node j, the message requests $VC_{1,+0}$ again until node k is reached.

3.4. Deadlock-freeness of Algorithm 3.1

In Algorithm 3.1, we have the following facts:

1. EW, WE, SN and NS messages use disjointed sets of virtual channels,
2. row messages (EW and WE) can become column messages (NS and SN), but no column messages can become row messages and

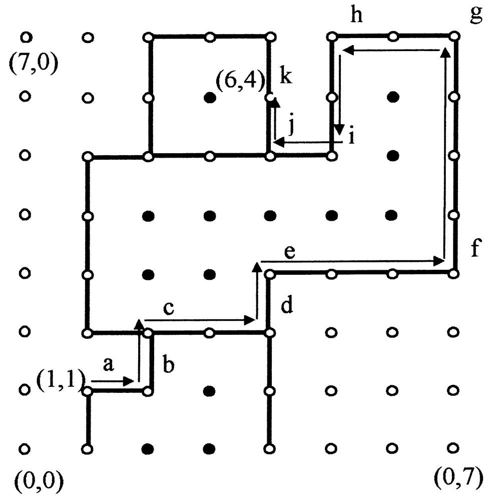

Fig. 4. A routing example from node *a* to node *k*.

3. EW and WE messages cannot change into each other and NS and SN cannot change into each other.

Thus, Algorithm 3.1 is a deadlock-free routing method if we can prove no deadlock will occur in each of the four message types.

Theorem 1. *Algorithm* 3.1 *is a deadlock-free routing method for two-dimensional meshes with convex faults.*

Proof. For a deadlock to occur among some messages, there must be a circular-wait on these messages. The circular-wait may occur within one column, one row or on different columns and rows in two-dimensional meshes. In the following, we show that no circular-wait can occur among row or column messages in these three situations.

Since no WE (resp. EW) row messages can travel from East to West (resp. West to East), no circular wait can occur among WE (resp. EW) row messages within one row or on different columns and rows. For a circular-wait to occur among row messages within one column, there must be two row messages such that each of the row messages has performed a U-turn and is waiting for the virtual channel occupied by the other. Since the length of each convex fault is smaller than its corresponding dimension length and no concave faults exist after fault-shrink, the condition for a circular-wait to occur among row messages within one column can never be satisfied. From the above statement, it can be concluded that no deadlock will occur among row messages.

NS (resp. SN) messages never perform a U-turn while heading to South (resp. North), no circular-wait can occur among column messages within one column. Although NS column messages, while routing on a row along a *f*-boundary, may perform a U-turn and reverses its routing direction, no circular-wait can occur among NS messages on a row because of the following channel requesting relationships:

1. an NS message routing on $VC_{0,-1}$ may depend on an NS message routing on $VC_{0,-1}$ or $VC_{0,+1}$;
2. an NS message routing on $VC_{0,+1}$ may depend on an NS message routing on $VC_{0,+1}$ or $VC_{0,-3}$;
3. an NS message routing on $VC_{0,+1}$ never depends on an NS message routing on $VC_{0,-1}$;
4. an NS message routing on $VC_{0,-3}$ never depends on an NS message routing on $VC_{0,+1}$;

By a symmetric argument, no circular-wait can occur among SN messages on a row only. Since (1) NS (resp. SN) messages always heading to South (resp. North) when they are not on *f*-boundaries and (2) no NS (resp SN) message can ever be routed on EMAX or EMIN nodes of any *f*-boundary while they are heading to North (resp. South), no circular-wait can occur among NS (resp. SN) column messages on different rows and columns.

From above statement, no circular-wait can occur among messages within one column, one row or on different columns and rows; therefore, Algorithm 3.1 is deadlock-free. □

Since (1) each row message can always get to the column on which its destination node sits, (2) each column message never was blocked by the same fault twice and (3) the number of convex faults is finite, Algorithm 3.1 is also livelock-free.

4. Adaptive fault-tolerant routing

4.1. Adaptive fault-tolerant routing algorithm

In Algorithm 3.1, messages are routed on virtual channels of class 0 if they are not blocked by faults. Virtual channels of classes 1, 2 and 3 of dimensions 0 and 1 on *f*-boundaries are intended and used for avoiding faults. All other virtual channels of classes 1, 2 and 3 are not used elsewhere. In order to improve network performance, these unused virtual channels are utilized to enhance Algorithm 3.1 to be an adaptive algorithm. The adaptability is introduced by allowing messages to use adaptively these unused virtual channels if the messages are not on the *f*-boundaries. Each adaptive routing step brings the message one step closer to its destination. Two more notations are defined.

$VC_{i,\alpha}$: the virtual channel of classes 0, 1, 2 and 3 of dimension i, $0 \leqslant i \leqslant 1$.
$VC_{i,\beta}$: the virtual channel of classes 1, 2 and 3 of dimension i, $0 \leqslant i \leqslant 1$.

Algorithm 4.1. Adaptive fault-tolerant routing algorithm for two-dimensional meshes with convex faults.

```
1. Calculate $R$;
2. If $R = 0$, route the message to local processor and exit;
3. If the message is not on f-boundary
   then if the message is a row message
        then if $v(R, 1) \neq 0$
             then parallel request $VC_{0,\alpha}$ and $VC_{1,\beta}$ and route the message on
             the first available virtual channel
                /* each routing step using $VC_{1,\beta}$ brings the message one step
                closer to its destination */
             else parallel request $VC_{0,\alpha}$ and route the message on the first avail-
             able virtual channel
        else parallel request $VC_{1,\alpha}$ and route the message on the first available
        virtual channel
   else route the message according to Algorithm 3.1;
```

When messages are not on *f*-boundaries, row messages parallel request $\mathrm{VC}_{0,\alpha}$ and $\mathrm{VC}_{1,\beta}$, if $v(R,1) \neq 0$, or otherwise request $\mathrm{VC}_{0,\alpha}$ only; column messages request $\mathrm{VC}_{1,\alpha}$. Each routing step on $\mathrm{VC}_{1,\beta}$ brings the row message one step closer to its destination. Messages are routed according to Algorithm 3.1 when they are on *f*-boundaries. The sets of virtual channels used by different message types are not disjointed in Algorithm 4.1.

4.2. Deadlock-freeness of Algorithm 4.1

Mesh networks are decomposed into two virtual interconnection networks, VIN1 and VIN2, to facilitate our discussion. VIN1 contains all the virtual channels of class 0 and all the virtual channels on *f*-boundaries of classes 1, 2 and 3; VIN2 contains all the other virtual channels. That is, messages can be routed in VIN1 or VIN2 when they are not on *f*-boundaries and can only be routed in VIN1 when they are on *f*-boundaries.

Theorem 2. *Algorithm* 4.1 *is a deadlock-free routing algorithm.*

Proof. Since (1) messages of different types use disjointed sets of virtual channels when they are routed in VIN1 and (2) there is always one virtual channel of class 0, which is in VIN1, can be requested at each routing step by messages routing in VIN2, messages of each particular type can only be blocked by messages of the same type in VIN1.

Since (1) no deadlock can occur among each of the message types in VIN1, for Algorithm 3.1 is deadlock-free and (2) routing in VIN2 never changes a column message into a row message and never bring a message to a node visited before, no deadlock can occur in each of the message type. From above statement, Algorithm 4.1 is deadlock-free. □

5. Conclusion

Designing a deadlock-free routing algorithm that can tolerate unlimited number of faults of arbitrary shape with constant number of virtual channels is not a easy job. Faulty blocks are expanded, by disabling good nodes, to be rectangular faults in existing literature to facilitate the designing of deadlock-free routing algorithms for mesh networks recently. The more the good nodes were disabled, the less the computation and transmission power was left. In this paper, for the purpose of reducing the number of good nodes disabled, we proposed a method to shrink, by recovering fault-diffused nodes, these rectangular faults. Only convex faults exist after fault-shrink. The simulation results show that up to 70% of the disabled nodes, which are needed to form rectangular faults, can be recovered if the number of original faulty nodes is less than 10% of the total network nodes.

We also proposed a non-adaptive and an adaptive routing algorithm to handle these convex faults. The adaptive algorithm is enhanced from the non-adaptive counterpart by utilizing the virtual channels that are not used in the non-adaptive algorithm. The method we used to enhance the non-adaptive algorithm is simple, easy and its principle is similar to the Duato's theorems [15,16]. There is no restriction on the number of faults tolerated and only four virtual channels per physical channel are needed in the proposed algorithms.

References

[1] P. Kermani, L. Kleinrock, Virtual cut-through: a new computer communication switching technique, Computer Networks 3 (1979) 267–286.

[2] K.D. Gunther, Prevention of deadlocks in packet-switched data transport systems, IEEE Transactions on Communications 29 (1981) 512–524.

[3] W.J. Dally, C.L. Seitz, Deadlock-free message routing in multiprocessor interconnection networks, IEEE Transactions on Computers 36 (5) (1987) 547–553.

[4] C.L. Seitz, W.C. Athas, C.M. Flaig, A.J. Martin, J. Seizovic, C.S. Steele, W.K. Su, The architecture and programming of the Ametek Series 2010 multicomputer, in: Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications, I, 1988, pp. 33–36.

[5] Intel Corp., A Touchstone DELTA System Description, 1991.

[6] D.H. Linder, J.C. Harden, An adaptive and fault-tolerant wormhole routing strategy for *k*-ary *n*-cubes, IEEE Transactions on Computers 40 (1) (1991) 2–12.

[7] A.A. Chien, J.H. Kim, Planar-adaptive routing: low-cost adaptive networks for multiprocessors, in: Proceedings of the International Symposium on Computer Architecture, 1992, pp. 268–277.

[8] L.M. Ni, P.K. Mckinley, A survey of wormhole routing techniques in direct networks, IEEE Computer 26 (1993) 62–76.

[9] W.J. Dally, H. Aoki, Deadlock-free adaptive routing in multicomputer networks using virtual channels, IEEE Transactions on Parallel and Distributed Systems 4 (4) (1993) 466–475.

[10] R.V. Boppana, S. Chalasani, Fault-tolerant routing with non-adaptive wormhole algorithms in mesh networks, in: Proceedings of Supercomputing, 1994, pp. 693–702.

[11] S. Chalasani, R.V. Boppana, Adaptive fault-tolerant wormhole routing algorithms with low virtual channel requirements, in: Internatinsl Symposium on Parallel Architecture, Algorithms, and Networks, 1994, pp. 214–221.

[12] R.V. Boppana, S. Chalasani, Fault-tolerant wormhole routing algorithms for mesh networks, IEEE Transactions on Computers 44 (7) (1995) 848–864.

[13] P.H. Sui, S.D. Wang, Comments on fault-tolerant wormhole routing algorithms for mesh networks, Technical Report NTUEE-TR-96-001, Department of Electrical Engineering, National Taiwan University, July 1996.

[14] C.C. Su, K.G. Shin, Adaptive fault-tolerant deadlock-free routing in meshes and hypercubes, IEEE Transactions on Computers 45 (6) (1996) 666–682.

[15] J. Duato, A new theory of deadlock-free adaptive routing in wormhole networks, IEEE Transactions on Parallel and Distributed Systems 4 (12) (1993) 1320–1331.

[16] J. Duato, A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks, IEEE Transactions on Parallel and Distributed Systems 6 (10) (1995) 1055–1067.

[17] P.H. Sui, S.D. Wang, Comments on adaptive fault-tolerant deadlock-free routing in meshes and hypercubes, Technical Report NTUEE-TR-97-001, Department of Electrical Engineering, National Taiwan University, April 1997.